

## **REMARKS**

In the Office Action, the Examiner rejected Claims 1-20, which were all of the then pending claims, under 35 U.S.C. 102 as being fully anticipated by U.S. patent application publication no. 2004/021876 (Stall, et al.). Claims 1-8 were further rejected under 35 U.S.C. 101 as directed to non-statutory subject matter. The Examiner also noted an informality in the specification and requested correction thereof.

Independent Claims 1 and 15 are being amended to better define the subject matters of these claims. Independent Claim 9 is being cancelled, and new independent Claim 21 is being added as a substitute. Claims 10-12 are being amended to be dependent from new Claim 21 instead of the now cancelled Claim 9 and to keep the language of Claims 10-12 consistent with the language of Claim 21. Claims 13 and 14 are being cancelled to reduce the number of issues in this case, and new Claim 22, which is dependent from Claim 21, is being added to describe an optional feature of this invention.

The specification is being amended to correct the informality noted by the Examiner. In particular, in paragraph 16, the spelling of "patterns" is being corrected. In view of this, the Examiner is requested to reconsider and to withdraw the objection to the specification.

The above-mentioned amendments also address the rejection of Claims 1-8 under 35 U.S.C. 101. To elaborate, the Examiner noted, in the Office Action, that these claims are directed to a software program, but were not associated with any physical structure.

In response, independent Claim 1 is being amended to set forth positively a physical medium and to indicate expressly that the software tool defined by Claim 1 contains machine readable instructions stored in that physical medium and executable by the machine to perform a specific function - that is, monitoring the behavior of a running computer program. As now

amended, Claim 1, and Claims 2-8, which are dependent from Claim 1, define an article of manufacture within the meaning of 35 U.S.C. 101 and are thus directed to statutory subject matter. The Examiner is, accordingly, also asked to reconsider and to withdraw the rejection of Claims 1-8 under 35 U.S.C. 101.

In addition, all of Claims 1-8, 10-12 and 15-22 patentably distinguish over the prior art because the prior art does not disclose or suggest the use of the pattern detectors as described in independent Claims 1, 15 and 21. The Examiner is thus asked to reconsider and to withdraw the rejection of Claims 1-8, 10-12 and 15-20 under 35 U.S.C. 102, and to allow these claims and new Claims 21 and 22.

In order to best understand this difference between the claims and the prior art, it may be helpful to review briefly the present invention and that prior art.

The instant invention, generally, relates to the automatic detection of problematic coding patterns and violations of best practices patterns at program runtime. As discussed in detail in the present application, in any large software deployment, subtle coding defects can cause problems in successful deployment of the software. Tracking down these defects may be extremely difficult in the production environment because of a number of factors.

One factor is that, in many cases, the symptoms are usually not unique to a particular type of software defect. This makes correlating symptoms to specific defects nearly impossible. Another factor is that many symptoms may manifest themselves only during production level loads and production configurations. This means that the defects are often undetected in the testing and debugging of software. In addition, the reasons why a piece of code is defective are often complex and may span third party libraries and frameworks.

The present invention effectively addresses these issues. Generally, this is done by observing the behavior of a running program within the context of a large number of defined coding patterns, and automatically flagging violations of the coding patterns when they occur. More specifically, in accordance with one aspect of the invention, a software tool is provided for monitoring the behavior of a running computer program for code patterns that violate a given set of coding rules. This software tool comprises a pattern detector manager and a plurality of individual pattern detectors.

The pattern detector manager is provided for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns. Each of the pattern detectors is also associated with one of the defined coding patterns, and each of the pattern detectors is invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with the pattern detector, is reached in the computer program. When invoked, a pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.

The prior art does not disclose or suggest the use of plural pattern detectors in the manner described above.

For example, Stall, et al. discloses a debugging procedure that is designed to skip over code that is not of interest during the debugging operation. There are a number of important differences between the preferred embodiment of the present invention and the procedure described in Stall, et al.

The focus of the present invention is the use of a debugger to automatically detect code patterns that violate a given set of coding rules. An example of a coding rule is: an invocation of method A should never follow an invocation of method B in a calling sequence. Such rules are generally impossible to verify prior to program runtime, because a compiler may not be able to enumerate all possible control flow paths through a static analysis of the program code. It is also impractical to enforce such a coding rule as a programming discipline. This invention provides a novel way of using a debugger (a tool that is conventionally used to determine the cause of a program bug, as in Stall et al.'s work) as an automatic verification tool to check when such coding rules are being violated, and to show the programmer the flow of control that created the violation.

Unlike debugging techniques that rely on manual insertion of breakpoints to pause execution of a running program in order to inspect some state, the present invention uses automatic breakpoint insertion to perform automatic checking of coding rules on the fly while the program is executing. The technique of this invention thus differs substantially from prior art that uses debugging techniques such as multiple breakpoint insertion for manual inspection of state for purposes of understanding the source of a program bug (which is described in Stall et al.'s patent).

Independent Claims 1, 15 and 21 describe important features not shown in or suggested by the prior art, and in particular, these claims describe the use of the pattern detector manager and the plurality of pattern detectors. The pattern detector manager is described in these claims as being used for inserting into the running computer program a plurality of entry breakpoints, each of which is associated with one of a plurality of defined coding patterns. Also, as described in Claims 1, 15 and 21, each of the pattern detectors is associated with one of the defined coding

patterns, and each of the pattern detectors is invoked by the pattern detector manager, after one of the entry breakpoints associated with the coding patterns that, in turn, is associated with that pattern detector, is reached in the computer program. These claims positively describe the further feature that, when invoked, a pattern detector determines whether the computer program violates the coding pattern associated with the pattern detector.


The other references of record have been reviewed, and these other references, whether considered individually or in combination, also do not disclose or suggest the use of a pattern detector manager and a plurality of pattern detectors, as described above.

Because of the above-discussed differences between Claims 1, 15 and 21 and the prior art, and because of the advantages associated with those differences, Claims 1, 15 and 21 patentably distinguish over the prior art and are allowable. Claims 2-8 are dependent from Claim 1 and are allowable therewith; and Claims 10-13 and 22 are dependent from, and are allowable with, Claim 21. Likewise, Claims 16-20 are dependent from Claim 15 and are allowable therewith. Accordingly, the Examiner is asked to reconsider and to withdraw the rejection of Claims 1-8, 10-13 and 15-20 under 35 U.S.C. 102, and to allow these claims and new Claims 21 and 22.

For the reasons set forth above, the Examiner is respectfully requested to reconsider and to withdraw the objection to the specification, the rejection of Claims 1-8 under 35 U.S.C. 101, and the rejection of Claims 1-8, 10-13 and 15-20 under 35 U.S.C. 102, and the Examiner is asked to allow Claims 1-8, 12-13 and 15-33.

If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully submitted,

  
John S. Sensny  
Registration No. 28,757  
Attorney for Applicants

Scully, Scott, Murphy & Presser, P.C.  
400 Garden City Plaza – Suite 300  
Garden City, New York 11530  
(516) 742-4343

JSS:jy